Cocoa Advanced View Techniques

Session 146

These are confidential sessions—please refrain from filming or taking pictures.

Application Technologies

Cocoa Advanced View Techniques

Getting the most from Tiger

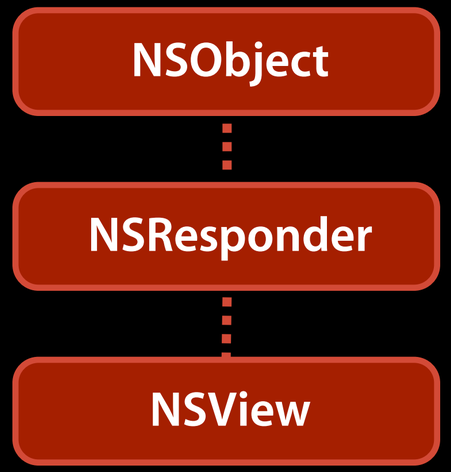
Troy Stephens
Application Frameworks Engineer

http://developer.apple.com/wwdc2005/

Views

The basic building blocks of window content

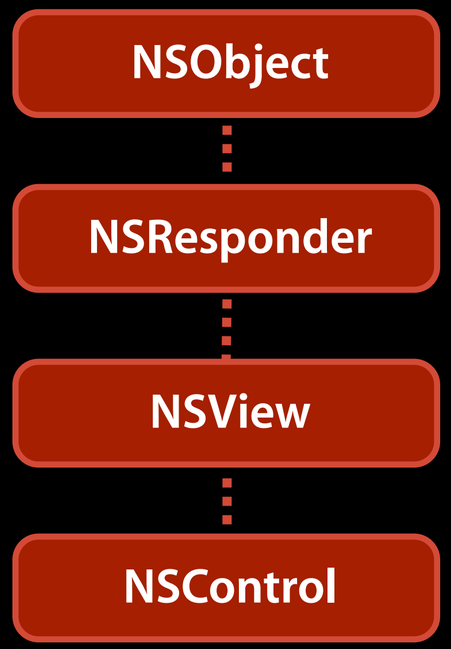
- Your application's interface to its users
- "Rectangles of responsibility"
  - Display content
  - Handle user input (mouse, keyboard events)
  - Nestable
  - Support coordinate transforms (translation, scaling, rotation)
  - Participate in the Responder chain
  - Support drag-and-drop
  - Part of a unified drawing/printing architecture
  - …and lots more! Tool tips, cursor management, tracking rects…



Views

The basic building blocks of window content

- Controls are views, too!
  - Delegate drawing, event-handling responsibility to an associated cell(s)
  - Can have an integer “tag”

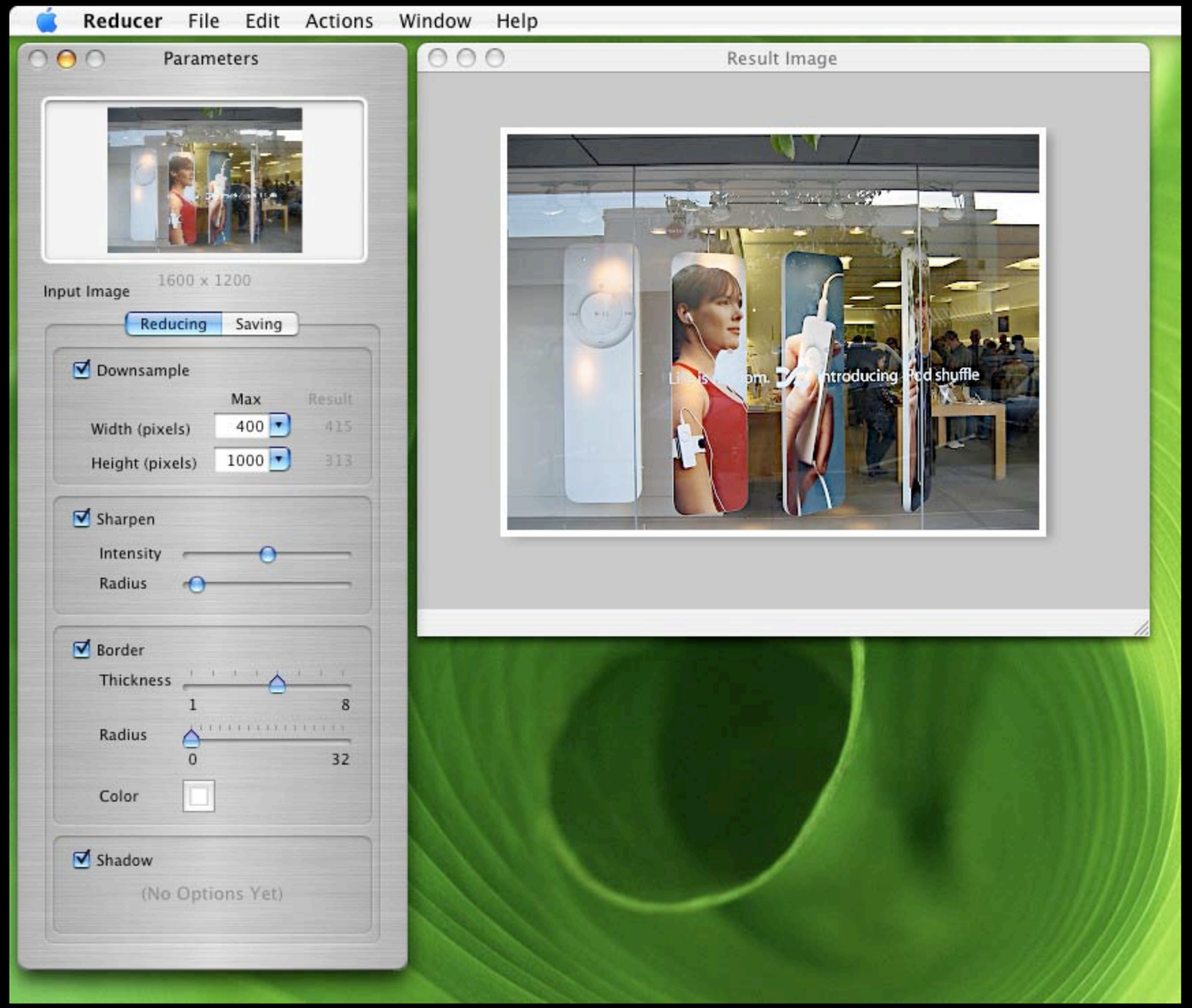


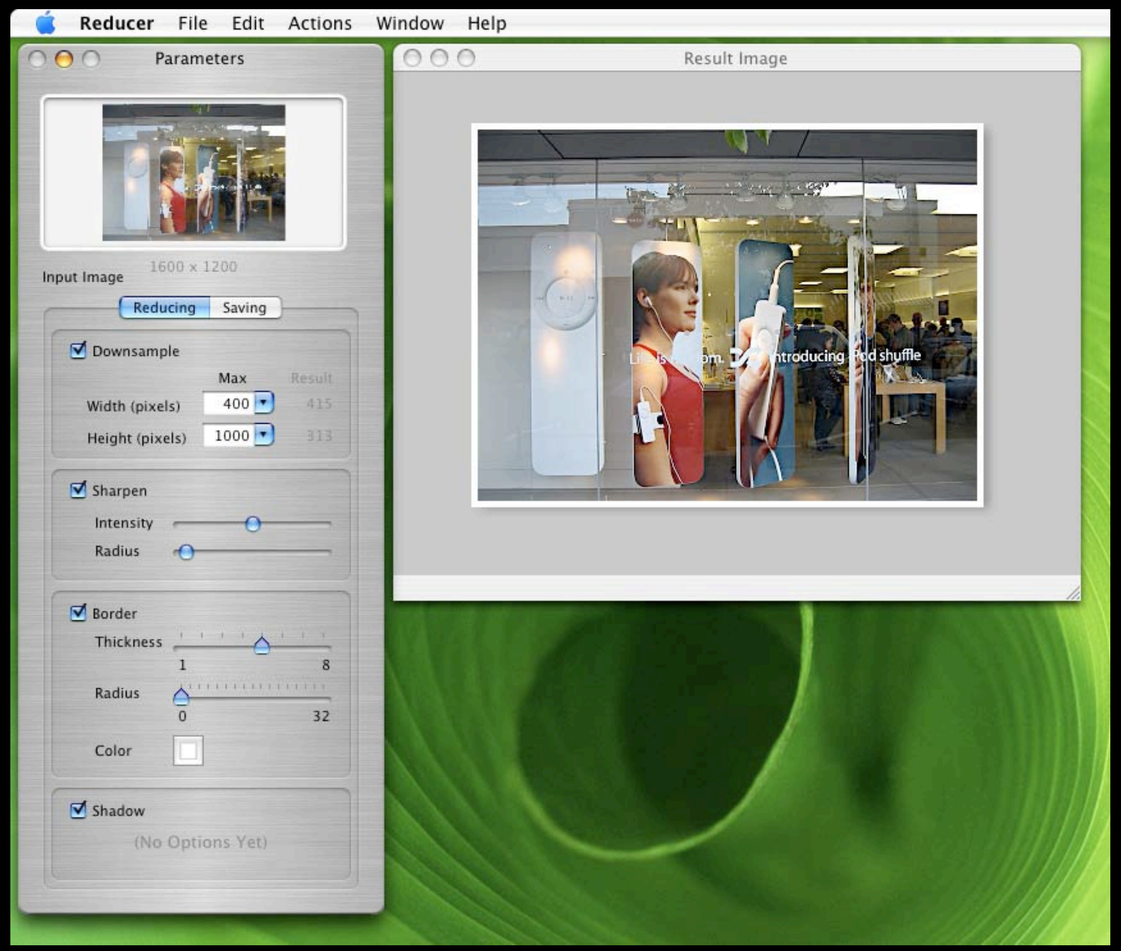
Introducing iPod shuffle
1600 X 1200

320 X 240

Today's Sample Project

"Reducer": a small image processing tool





Sample Project

Features

- Downsampling to fewer pixels
- Sharpening
- Adding a border
- Adding a drop shadow
- Saving the result as JPEG, PNG, TIFF, or BMP
- Composing an XHTML `<img .../>` tag

What We'll Cover

- Using NSAnimation to liven up view state changes

- Leveraging the power of CoreImage in a Cocoa application

- Capturing view drawing using NSView "drawing redirection" methods

- Using an NSOpenGLContext with an ordinary NSView
- Creating a custom view class
  - Putting the view in an Interface Builder palette
  - Making the view's attributes bindable in Interface Builder

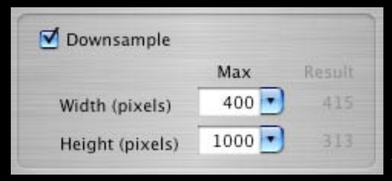
Demo

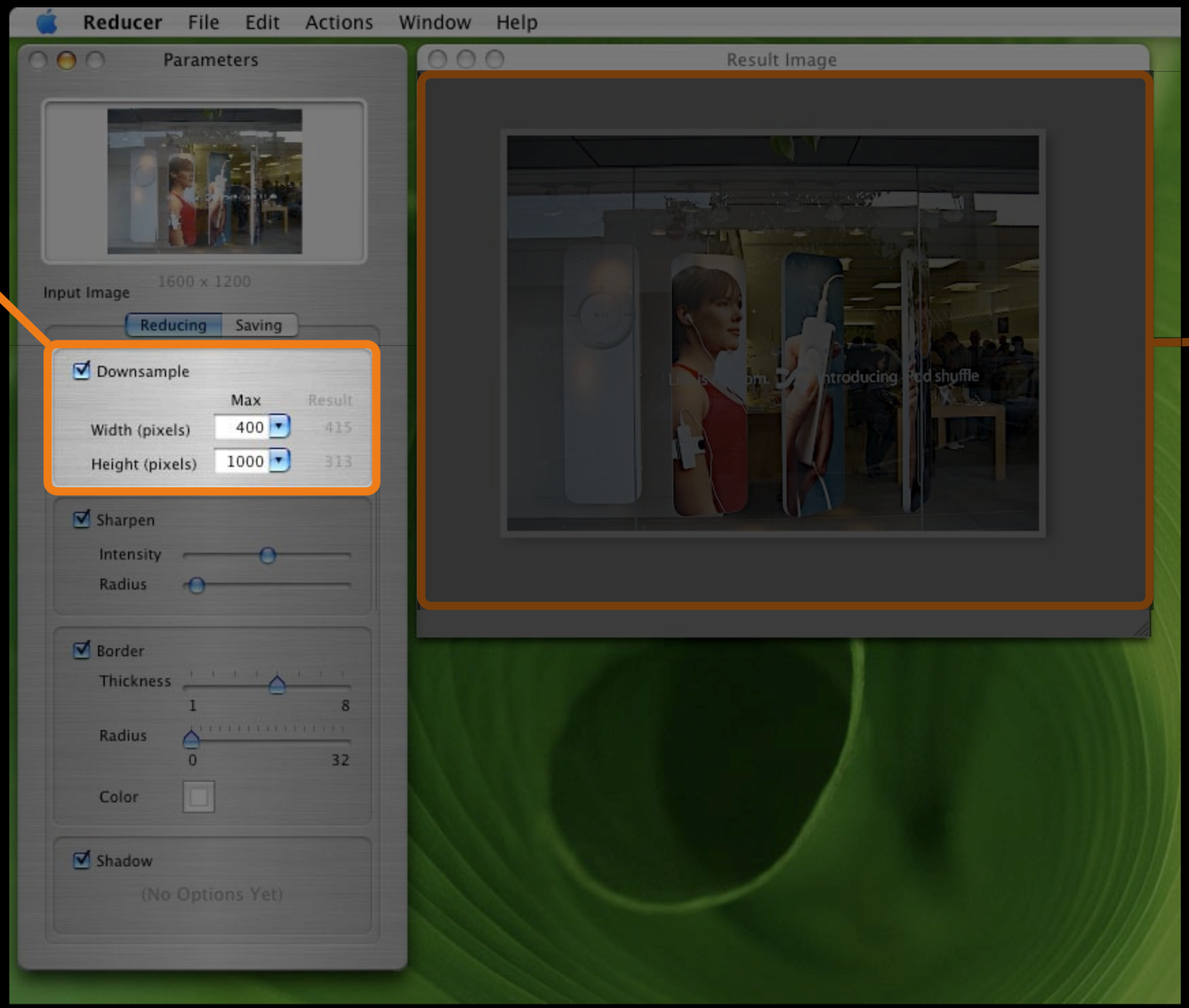
The “Reducer” Application

http://developer.apple.com/wwdc2005/

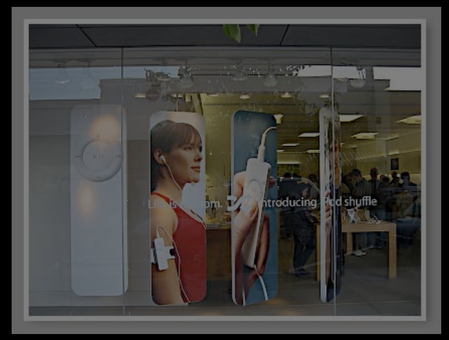
Custom Views for Today's Sample Project

CollapsibleBox





HybridImageView



Part 1. Making a Collapsible Box

http://developer.apple.com/wwdc2005/

Creating a Custom View Class

- Why do it?
  - To customize the appearance or behavior of an existing view or control
  - To define a brand-new NSView subclass for displaying and interacting with application-defined data
- We'll look at examples of both today

Implementing a "CollapsibleBox" View

- Subclass NSBox to inherit its nice border-drawing functionality
- And add an "expanded" Boolean attribute
- Implement KVC-compliant accessors for "expanded" attribute
  - Have `-isExpanded` report the current state
  - Have `-setExpanded:` remember the current frame size, and change the view's frame to match the new state
- Animate the transition to the new frame size

NSAnimation Classes

NSAnimation.h

NSAnimation

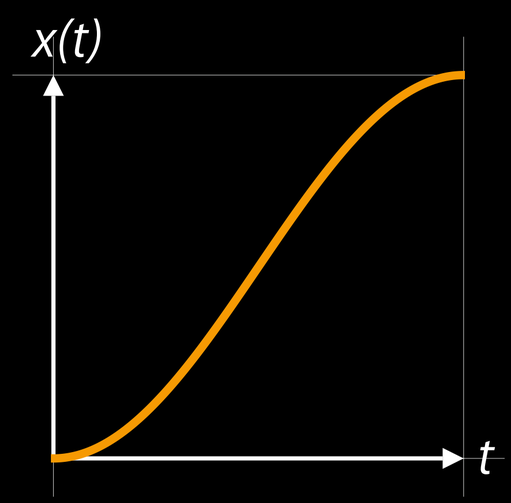
- Timing
- Start/Stop control and chaining
- Animation progress curves - *x(t)*
- Delegate mechanism

NSViewAnimation

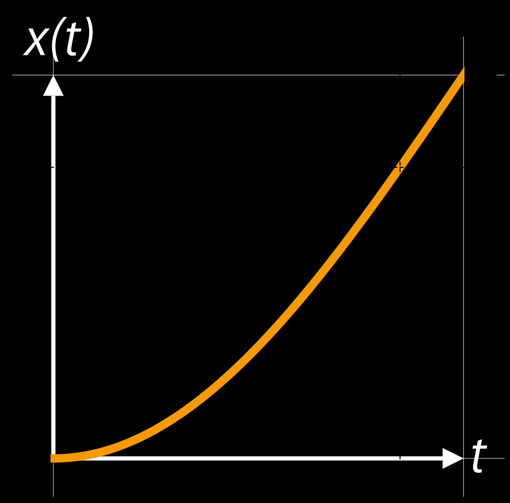
- Works with views and windows
- Move views around
- Animate view sizes, stretching and shrinking content
- Fade views in and out

Standard Animation Curves

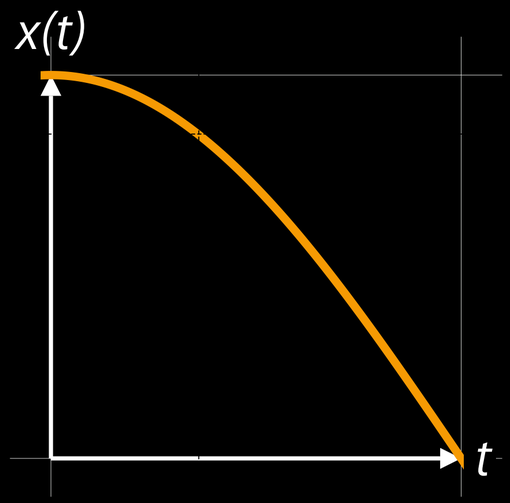
NSAnimation.h



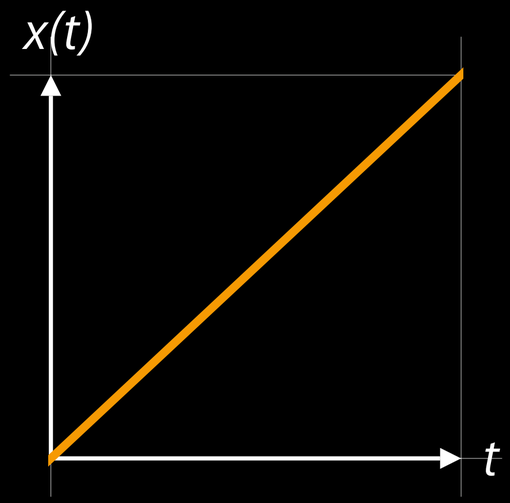
NSAnimationEaseInOut



NSAnimationEaseIn



NSAnimationEaseOut



NSAnimationLinear

- Sheets
- Drawers

Animating One or More Views

- Build the animation description (an NSArray of NSDictionary instances)
- Create an NSViewAnimation instance from the description
- Set duration, blocking mode, and optional delegate
- Start the animation
- Release the animation when done

Animating One or More Views

1. Assemble the list of animation descriptions, one per target

```
NSMutableArray *animationArray = [NSMutableArray arrayWithObjects:

    [NSDictionary dictionaryWithObjectsAndKeys:
        self,                                   NSViewAnimationTargetKey,
        [NSValue valueWithRect:oldFrame], NSViewAnimationStartFrameKey,
        [NSValue valueWithRect:newFrame], NSViewAnimationEndFrameKey,
        NSViewAnimationFadeOutEffect,     NSViewAnimationEffectKey,
        nil],

    // Add more animation description dictionaries here.

    nil
];
```

Animating One or More Views

2. Create an NSViewAnimation instance, and run the animation

```
NSAnimation *animation = [[NSViewAnimation alloc]
    initWithViewAnimations:animationArray];
[animation setAnimationBlockingMode:NSAnimationBlocking];
[animation setDuration:1.0];
[animation startAnimation];
[animation release];
```

Demo

Examining the “CollapsibleBox” Class

http://developer.apple.com/wwdc2005/

Animating a View Hierarchy

- Complex view tree animations might not “just work”
- As an alternative, you can use view autosizing with NSViewAnimation
  - Set desired autoresizing masks for views in the hierarchy
  - Drive the topmost view’s resize using an NSViewAnimation

Putting a View in a Palette

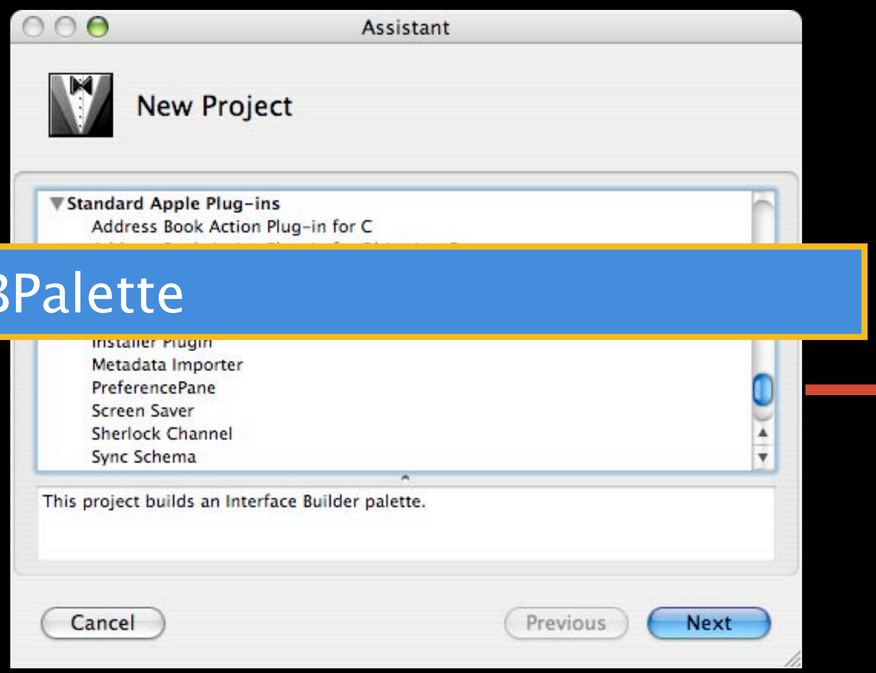
Why create a framework and palette for your custom view(s)?

- Make it easier to configure instances: visually in IB, using an inspector
- Views with a supporting palette are real, live objects in IB!
- Distribute your palettes of custom views, with the same IB-integration polish as standard AppKit views

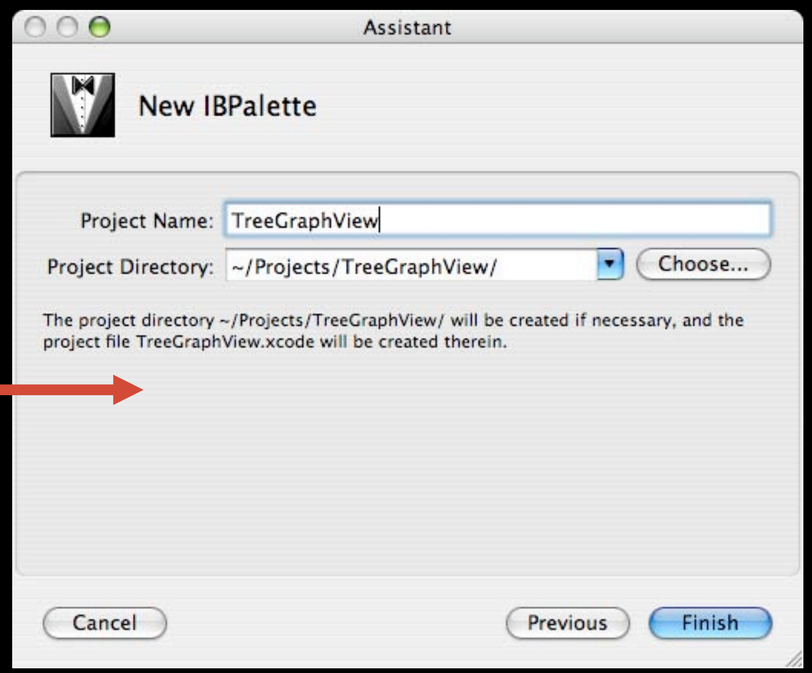
Creating an IB Palette Project in Xcode

File ➡ new project… ➡ standard Apple plug-ins ➡ IBPalette

Give the project the desired name of your view class (e.g. “CollapsibleBox”).



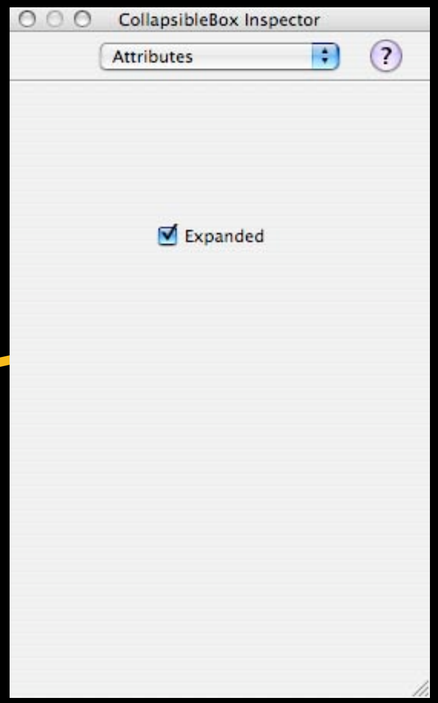
IBPalette

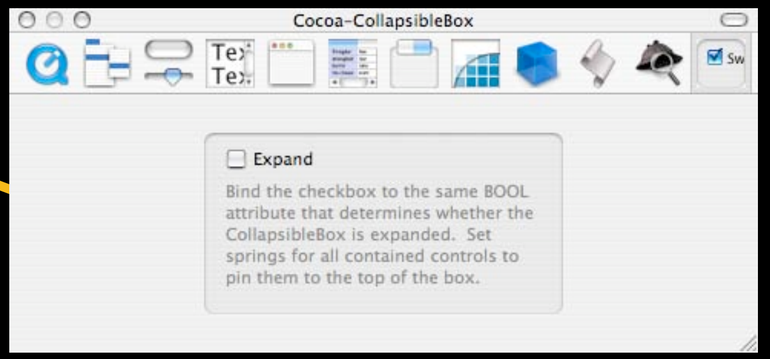


This creates a project with framework and palette targets.

Putting Your Custom View on a Palette

- In Xcode 2, this template creates a project with Palette and Framework targets
  - Framework
    - *MyView*
  - Palette
    - *MyView*Inspector
    - *MyView*Palette





Putting a View in a Palette

Further steps to make this work

- Required
  - Need to set classes for File's Owner in palette and inspector .nib files
  - Implement `-initWithCoder:` / `-encodeWithCoder:` for your view class
  - Implement setter accessors to do `[self setNeedsDisplay:]` as needed
  - Initialize your prototype view instance's attributes
- Optional polish
  - Build an inspector for your view's properties
  - Give the palette an icon (32 x 32 pixels)
  - Expose interesting attributes as bindable
  - Can use a placeholder in the palette instead of your view, if desired

Exposing Bindable Attributes

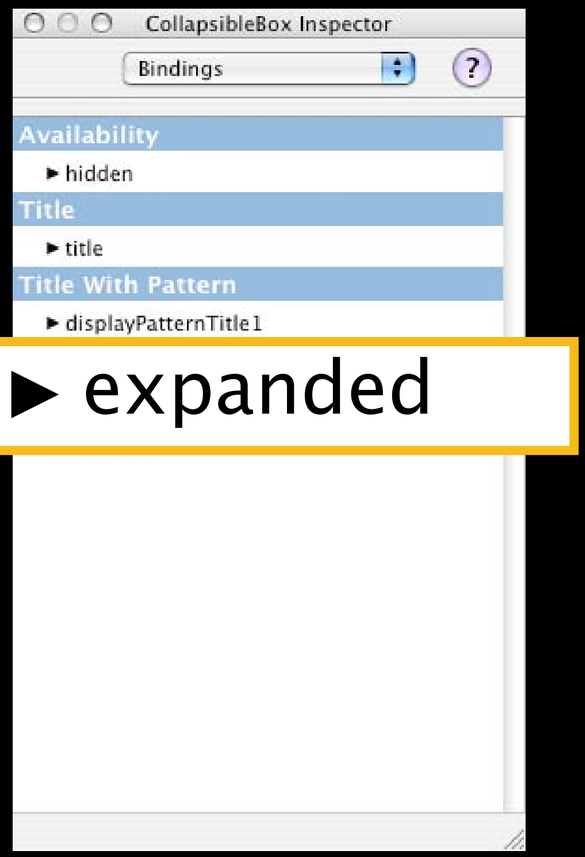
1. Provide KVC-compliant accessors

```
@implementation CollapsibleBox

- (BOOL) isExpanded {
    return expanded;
}

- (void) setExpanded:(BOOL)flag {
    if (expanded != flag) {
        expanded = flag;
        [self setFrameSize:otherFrameSize animate:YES];
    }
}

@end
```

Exposing Bindable Attributes in IB

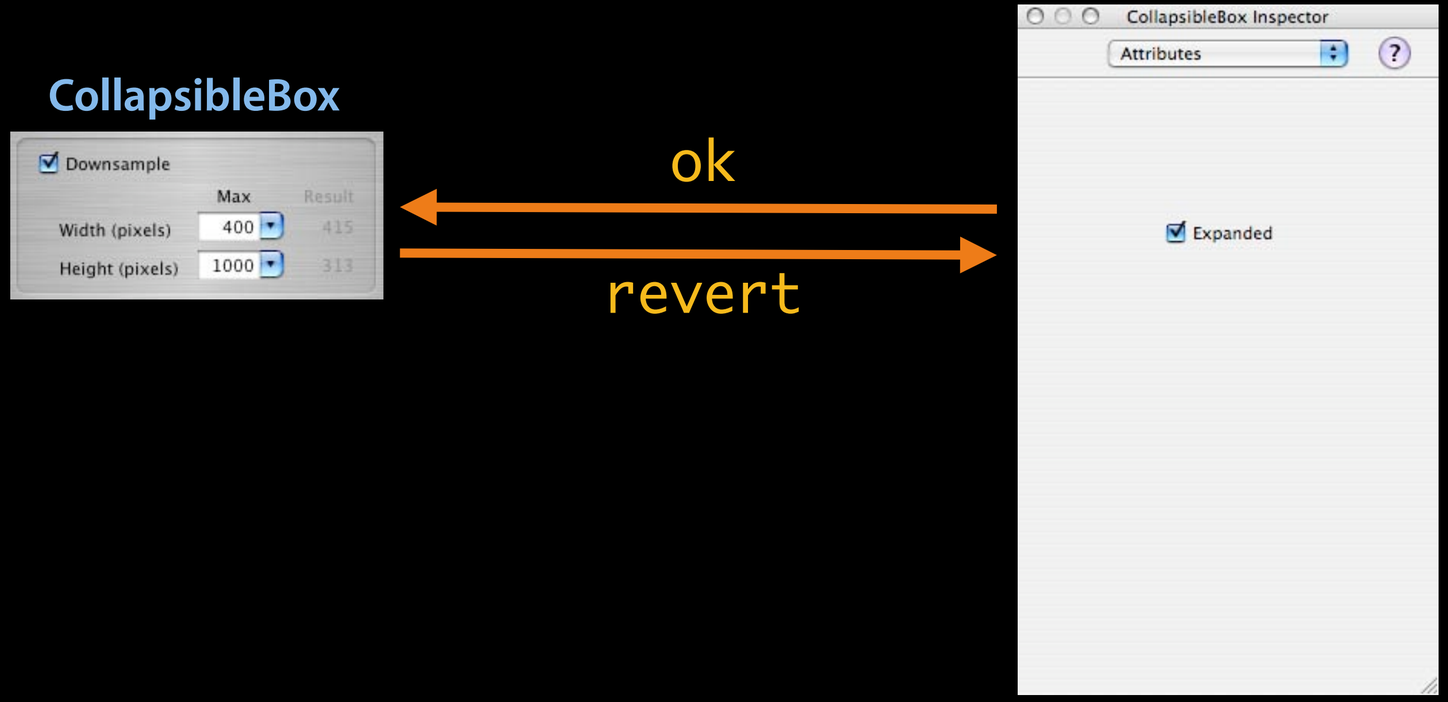
2. Declare each attribute (class `+initialize` is a convenient place)

```
@implementation CollapsibleBox

+ (void) initialize {
    if (self == [CollapsibleBox class]) {
        [self exposeBinding:@”expanded”];
        ...
    }
}

@end
```



Creating an Inspector For Your View

- Create outlets from "*MyView*Inspector" to its controls
- Wire your controls' actions to send "ok:" to the inspector
- Implement `-ok:` to pull controls' state into inspected view's properties
- Implement `-revert:` to push view's properties into controls' state



Finishing Prototype View Initialization

- Provide a common instance initialization method to be invoked when palette is initialized
- *MyView*Palette's -finishInstantiate is a good place to invoke this from

```
@implementation MyViewPalette

- (void)finishInstantiate {
    // Initialize the properties that MyView adds to its parent class.
    [myViewPrototype initAddedProperties];
}

...

@end
```

Demo

Examining the “CollapsibleBox” Palette

http://developer.apple.com/wwdc2005/

Part 2. A Hybrid Image View

Using OpenGL with AppKit Views

http://developer.apple.com/wwdc2005/

“ImageReducer” Class

Reducer’s image processing core

Parameters

ImageReducer

AppKit CoreImage Additions

Bridging the gap

NSColor.h

NSCIImageRep.h

AppKit

NSImageRep

NSBitmapImageRep

NSCIImageRep

```
+ (id)imageRepWithCIImage:(CIImage *)image;
- (id)initWithCIImage:(CIImage *)image;
- (CIImage *)CIImage;
```

NSColor

NSGraphicsContext

Core Image

CIImage

```
-drawInRect: fromRect:operation:fraction:
-drawAtPoint:fromRect:operation:fraction:
```

CIColor

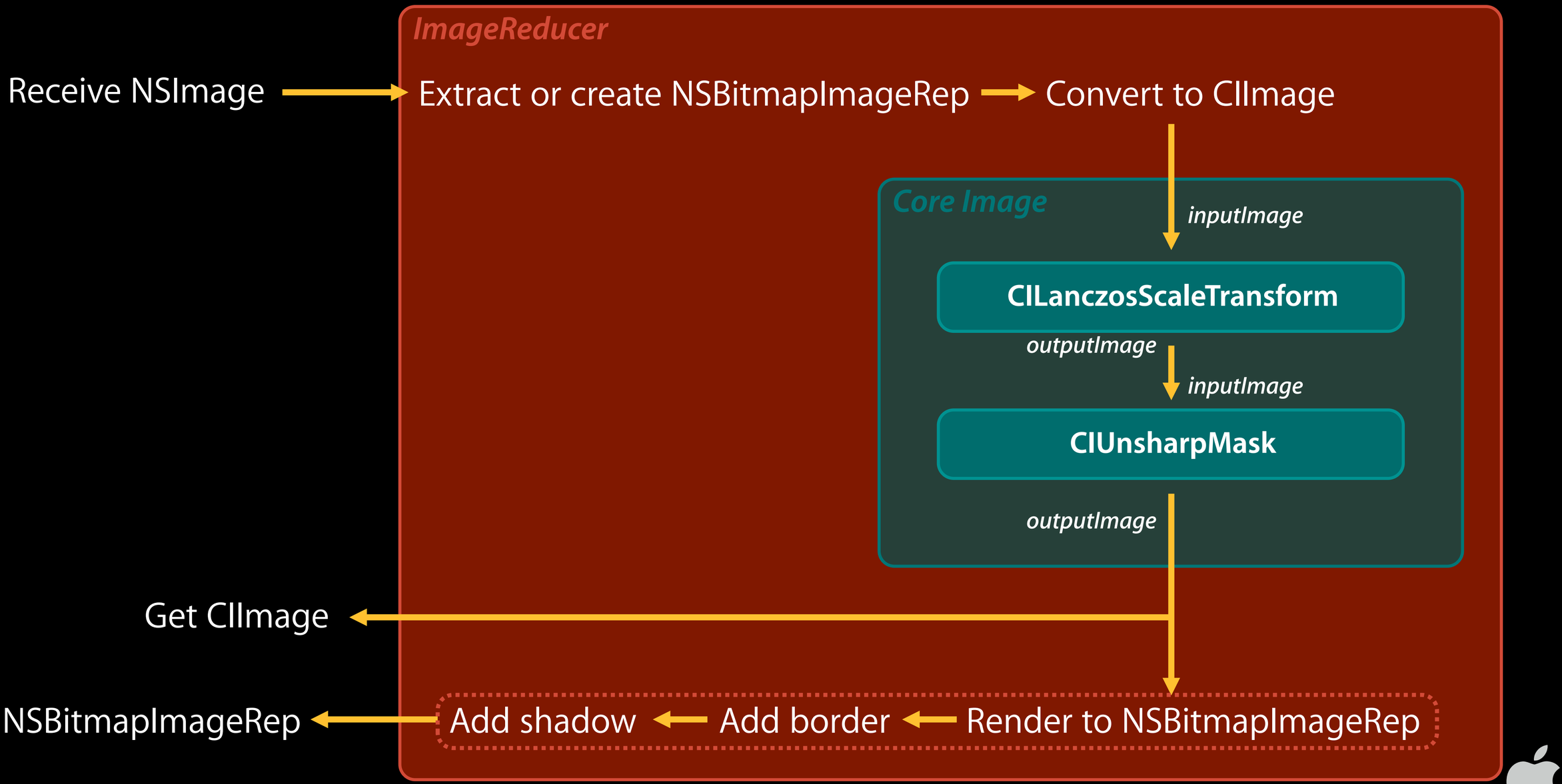
CIContext

-initWithColor:

+colorWithCIColor:

-CIContext

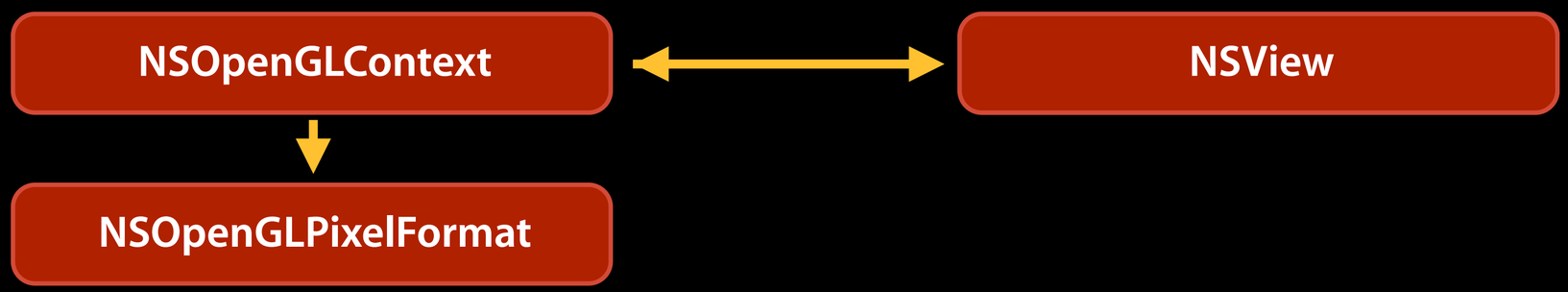
ImageReducer Class Internals



Implementing a “Hybrid” Image View

Using NSOpenGLContext with an ordinary NSView

```
- (void)setView:(NSView *)view;
- (NSView *)view;
```



Demo

“HybridImageView” in Action

http://developer.apple.com/wwdc2005/

Implementing a “Hybrid” Image View

Alternating between OpenGL and Quartz Drawing

- Switching to OpenGL drawing
  - Create an instance of NSOpenGLContext
  - Connect the context to the view using `-setView:`
  - Make the context current using `-makeCurrentContext`
  - Perform OpenGL initialization using the OpenGL API
  - Draw the view’s content using `gl...()` calls!
  - Optionally, send `[window disableScreenUpdatesUntilFlush]`
    - Prevents “flash” of surface content before it has been initialized by your app

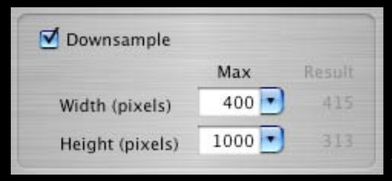
Implementing a “Hybrid” Image View

Alternating between OpenGL and Quartz Drawing

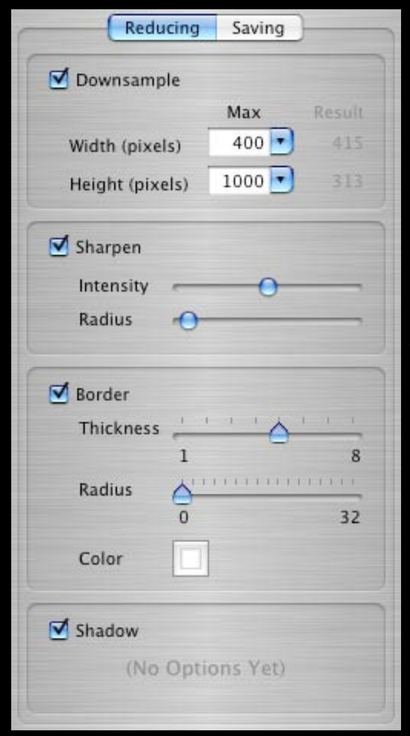
- Switching back to Quartz drawing
  - Dismiss the context’s surface using `-clearDrawable`
  - Release the NSOpenGLContext instance
  - Draw the view’s content using Cocoa and Quartz APIs.
  - Optionally, send `[window disableScreenUpdatesUntilFlush]`
    - Prevents “flash” of window backing store before it has been initialized

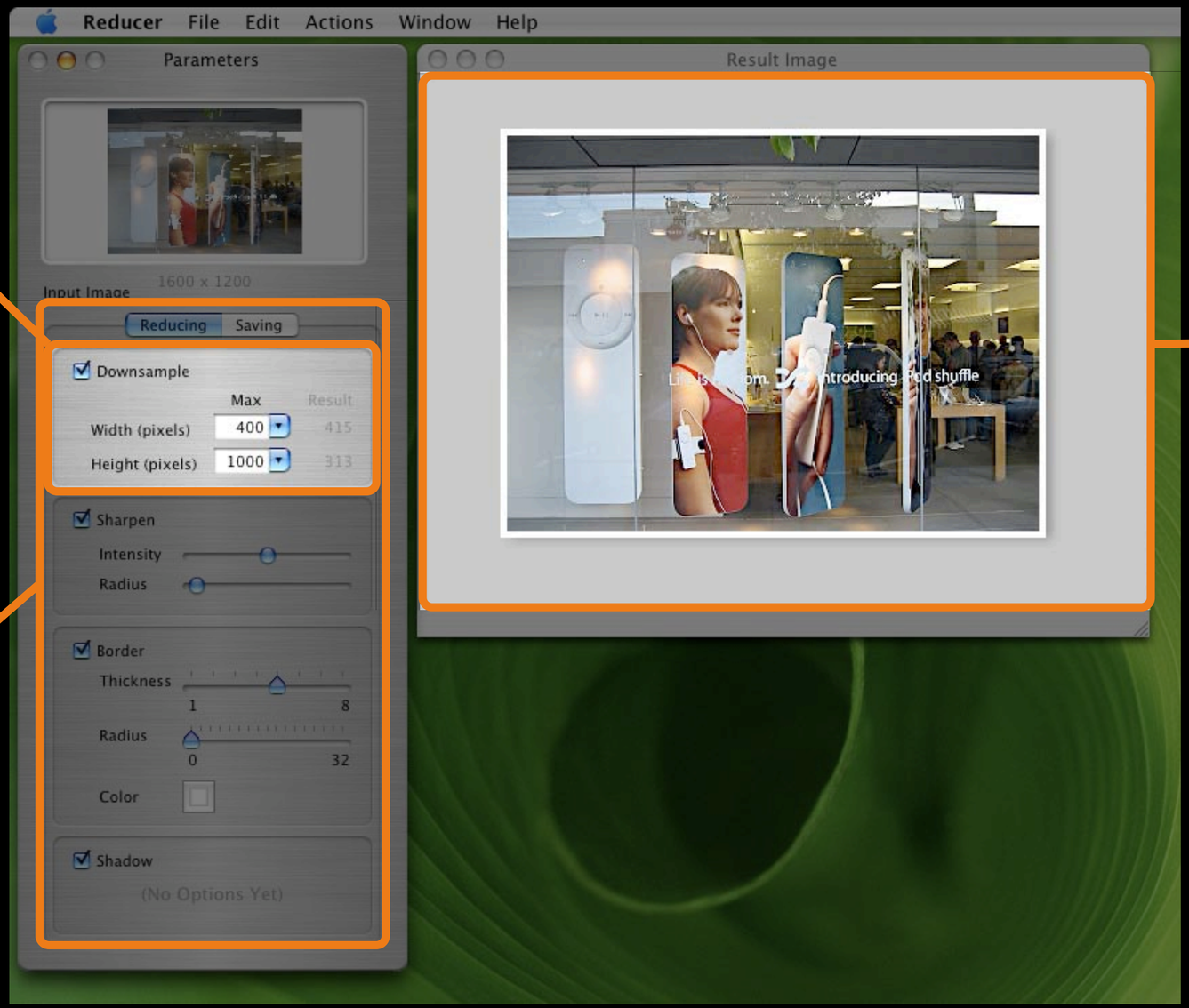
Custom Views for Today's Sample Project

CollapsibleBox



AnimatingTabView





HybridImageView

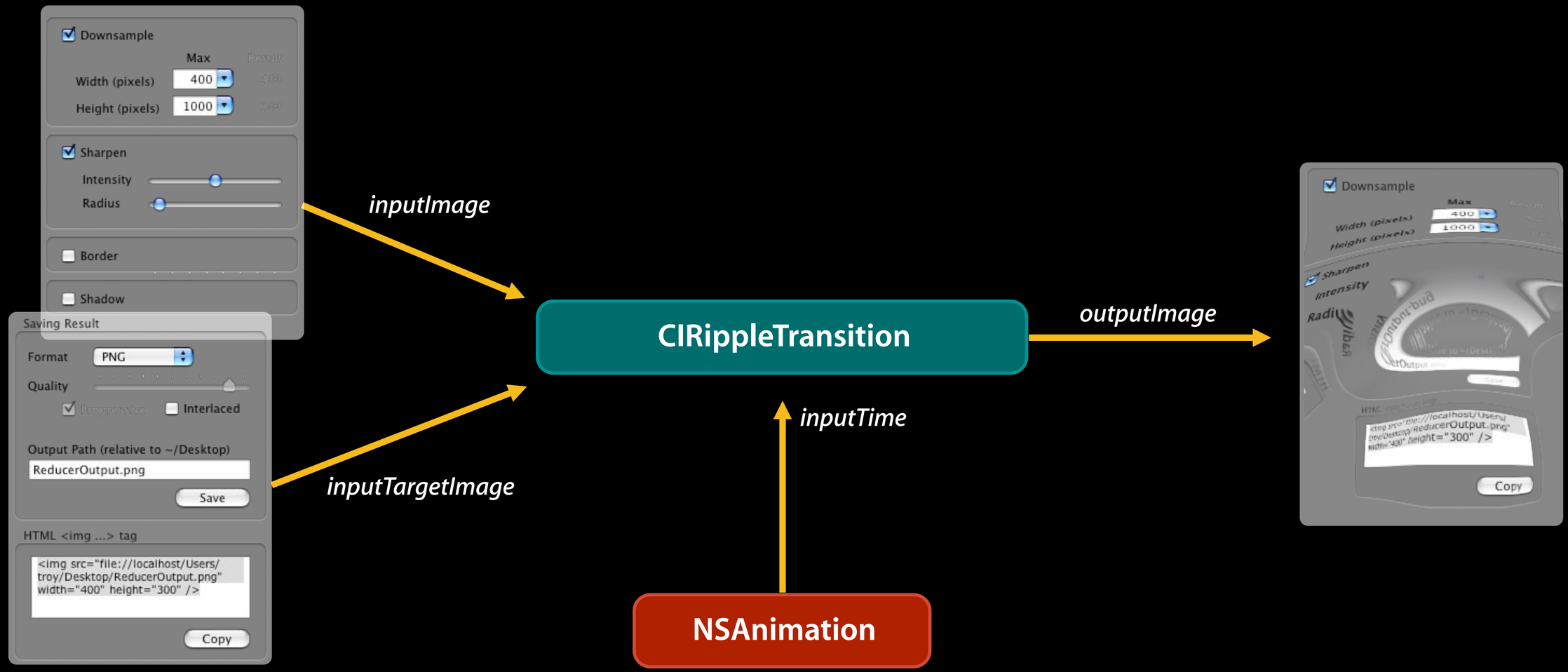
Part 3. Animating a Tab Switch

Using CoreImage with View Drawing Redirection

http://developer.apple.com/wwdc2005/

Animating a Tab Switch

View Drawing Redirection + Core Image + NSAnimation



Capturing View Content

Two approaches

Grab from Window Backing Store

```
NSRect rect = [view bounds];
[view lockFocus];
NSBitmapImageRep *bitmap =
    [[NSBitmapImageRep alloc]
    initWithFocusedViewRect:rect];
[view unlockFocus];
```

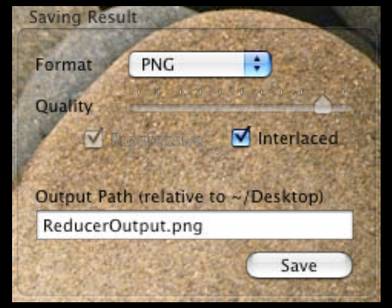
View must be in a window (offscreen OK)

Result includes window background

Draw View into a Bitmap new

```
NSRect rect = [view bounds];
NSBitmapImageRep *bitmapImageRep =
    [view bitmapImageRepForCachingDisplayInRect:rect];
[view cacheDisplayInRect:rect
   toBitmapImageRep:bitmapImageRep];
```

Window not necessary



Result can include transparent areas

Drawing "Redirection" Methods

NSView

High-Level

```
- (NSBitmapImageRep *) bitmapImageRepForCachingDisplayInRect:(NSRect) rect;
- (void) cacheDisplayInRect:(NSRect) rect
         toBitmapImageRep:(NSBitmapImageRep *) bitmapImageRep;
```

Funnel Point

```
- (void) displayRectIgnoringOpacity:(NSRect) aRect
                          inContext:(NSGraphicsContext *) context;
```

NSGraphicsContext

```
+ (NSGraphicsContext *)
    graphicsContextWithBitmapImageRep:(NSBitmapImageRep *) bitmapRep;
```

Demo

“AnimatingTabView”

http://developer.apple.com/wwdc2005/

Conclusion

http://developer.apple.com/wwdc2005/

Topics Covered

- Using NSAnimation to liven up view state changes

- Leveraging the power of CoreImage in a Cocoa application

- Capturing view drawing using NSView “drawing redirection” methods

- Using an NSOpenGLContext with an ordinary NSView
- Creating a custom view class
  - Putting the view in an Interface Builder palette
  - Making the view’s attributes bindable in Interface Builder

More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Who to Contact

Matthew Formica
Cocoa and Development Tools Evangelist
mformica@apple.com

Q&A

Ali Ozer
Cocoa Frameworks

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference